2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science

# Security Analysis of Open Source SDN (ODL and ONOS) Controllers

Pulkit Ohri
Research Scholar, Dept. of CSE
Amity University Madhya Pradesh
Gwalior, MP, India
pulkit.ohri@s.amity.edu

Subhrendu Guha Neogi
Associate Professor, Dept. of CSE
Amity University Madhya Pradesh
Gwalior, MP, India
sgneogi@gwa.amity.edu

Sunil Kumar Muttoo
Professor, Dept. of Computer Science
University of Delhi
New Delhi, Delhi, India
drskmuttoo@gmail.com

2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS) | 979-8-3503-9874-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/SCEECS57921.2023.10063108

***Abstract*—Software-Defined Networking (SDN) is becoming an increasingly relevant networking approach day by day, due to its myriad benefits over Traditional hardwired-based Networks. Still, SDN Controllers are popular for their weak security mechanism. This paper discusses the vulnerability issues in the security of SDN Controllers. The Control layer is the most vulnerable to network attacks in the three-layered SDN architecture. DDoS attacks on the Control layer can cause a Single Point of Failure for the entire SDN network. OpenDaylight (ODL) and Open Networking Operating System (ONOS) are the top two leading open-source SDN Controllers. We found out that ODL and ONOS Controller fail to provide any protection from DDoS attacks. SDN developers have compromised the security aspects of the Controller to improve its performance. In this research article, an attempt is being made to address such security vulnerabilities of SDN with ODL and ONOS Controllers for DDoS attacks.**

***Keywords—ODL, ONOS, Defense4All, Wireshark, Hping3***

## I. Introduction

Due to the exponential growth of cheap Internet and low-cost devices, the number of network devices on our planet is increasing rapidly [1]. This rapid development increases the workload on network servers, making it difficult to control, manage and update them using the traditional networking approach. When the network size has increased dramatically, the Traditional Networking approach will not be a feasible and optimal technique to use [2]. The Traditional Networking approach uses a decentralized control mechanism to control and update devices, which is optimal only in the case of a limited number of devices. SDN is a new centralized network management scheme that makes the network easily manageable and flexible to use.

SDN decouples the control plane from the application and data plane, allowing centralized control over the network devices [3]. Devices can now be controlled using a single click of a mouse due to the help provided by computer programming. SDN simplifies network complexity, integration of third-party applications becomes easy, eliminates manual configuration of networks, and helps in achieving Intent-Based Networking (IBN) [4]. Using SDN as a stepping-stone, organizations are moving towards the evolution of SDN, known as IBN. IBN can be seen as an evolution of SDN.

IBN expands SDN to automate the essential tasks in an organization. SDN automates several time-consuming tasks like re-configuration and scaling of the network. An Intent-Based network fully automates the task by using Artificial Intelligence (AI) to self-check the network. In case of any network change or error, IBN automatically notifies the administrator of the needed action or can fix the error by itself. IBN reduces manual intervention from the network administrator and provides increased security, robustness, and reduced expenditure on planning, testing, and manual configuration of network policies. SDN thus serves as a backbone for IBN [5] [6].

More than 20 SDN Controllers are available in the market. Some of them are RYU, POX, NOX, Floodlight Controller, Cherry, light Core, OpenVSwitch, Open Daylight (ODL), and Open Network Operating System (ONOS) Controller. Among them, ONOS and ODL contain the highest market share in the SDN market. ONOS stands first and ODL second in terms of market share. ODL and ONOS are used as a foundation by different vendors to create their own commercial SDN Controller. No controller is supreme, and both have their advantages and disadvantages. In this paper, we have comprehensively evaluated the robustness of ODL and ONOS controllers against web-based attacks [7].

## II. DDoS Protection Mechanism of ODL Controller

DoS/DDoS attack's main purpose is to make legitimate user requests unavailable to the particular infrastructure. Due to the Single Point of Failure of the SDN Control layer, these attacks remain the most fearful of all the network attacks currently available currently in SDN [8]. Defense4All is an anti-DDoS application, released by Radware Corporation to thwart DoS/DDoS attacks. Defense4All is the first and last line of defence against the DDoS attack. Radware Corporation integrated its DDoS mitigation application "Defense4All" with ODL Controller in 2014 [9]. Other security components in ODL Controller include Authentication, Authorization, accounting (AAA), Controller Shield, ODL-Cardinal, Secure Network Bootstrapping Interface, and more. Development of Defense4All stays at a snail's pace, and critical vulnerabilities like SQL Injection have been found in the same [10].

Defense4All works in two phases, namely the Attack Detection and Attack Mitigation Phase. In Attack Detection Phase, Defense4All detects the suspicious behaviour of all the hosts connected to the Controller. Defense4All monitors network traffic by installing traffic counting flows in the Openflow Switches. These switches then send back their flow table entries data to the Defense4All Application. Defense4All then monitors the traffic of all these switches. Then it analyzes the traffic and generates various baseline parameters like the average number of users accessing the server, the number of responses and replies sent to and from the server, and more. If a continuous significant deviation from the normal learned traffic baseline is detected, Defense4All immediately alerts the ODL Controller of this suspicious activity.

In the Second Phase, the DDoS attack is mitigated by redirecting the malicious traffic to some temporary hosts, also known as Attack Mitigating Hosts (AMC). Redirecting traffic makes the attacked hosts return to their normal flow of operation. Using these two steps, the DDoS attack is successfully mitigated by the attacked hosts. Suppose malicious traffic is generated from TCP Protocol. In that case, only TCP Protocol data is redirected to AMS, and the rest protocols, like UDP and SMTP, continue to work normally. Once the DDoS attack is over, this redirection and malicious flow rule entries are removed from the attacked host. Due to this technique, the ODL Controller automatically returns to its normal mode of operation once the DDoS attack is over.

## III. DDOS PROTECTION MECHANISM OF ONOS CONTROLLER

Like ODL Defense4All Application, ONOS does not provide any dedicated DDoS Detection and Mitigation Application. ONOS provides a security module called "Security Mode ONOS" [11] [12]. Security Mode ONOS is developed since external applications can use malicious or buggy code, steal data, and cause DoS attacks on the network. ONOS provides complete control of Northbound APIs and access to network resources for Third-party Applications. ONOS is an Application layer Access Control mechanism that protects ONOS from malicious applications. This freedom allows external applications to access any resources they intend to use. In real-world environments like Mission Critical Networks, the network administrator needs conservative access to protect sensitive data. Third-Party applications can also launch attacks on other hosts like data infiltration, network manipulation, DoS/DDoS, etc. They can also contain buggy code, which may install backdoors on the network server for gaining unauthorized network access.

Security-Mode ONOS provides two features, namely Application Authentication, and Permission-based Access Control, for protection against network attacks. Security Mode ONOS is disabled by default and needs manual configuration to run on the SDN Controller. In Security Mode ONOS, no module can detect or mitigate external DDoS attacks.

## IV. METHODOLOGY

We used the following software to set up the test-bed environment for our experiment. Mininet emulator tool was used to virtually create an environment of routers, switches, and controllers [13]. Mininet works on Linux Operating System and supports the Openflow protocol used by the SDN Controllers. Mininet provides an inexpensive, reconfigurable, and faster way to create networks on the fly. Separate Virtual Machines were used to install ODL and ONOS Controllers. ONOS and ODL Controllers are installed on separate virtual machines [14]. The software used in this study is as follows:

- Oracle VM Virtual Box
- ODL Controller VM
- ONOS Tutorial VM
- hping3 tool
- Slowloris tool
- Ubuntu 16.04 LTS VM (Mininet Emulator)

## V. EXPERIMENTAL ANALYSIS

In this section, we performed Penetration Testing of ODL and ONOS controllers. Fifteen different DDoS attacks were launched to check the robustness of these Controllers. Both ODL and ONOS fail to mitigate these DDoS attacks. We found out that ODL Controller is slightly more immune than ONOS Controller.

TABLE I. DETAILS OF VIRTUAL MACHINES

| **Virtual machine** | **Software Details** | **IP Address** | **Specification** |
|---|---|---|---|
| ONOS VM | ONOS Version 1.15.0 (Peacock) | 192.168.56.109 (VM IP) and 172.17.0.5 (ONOS Operating System IP) | Intel core i5 Processor with 5GB RAM |
| SDN Hub VM | ODL Version 0.6.0 (Carbon) | 192.168.56.102 | Intel core i5 Processor with 3GB RAM |
| Ubuntu 16.04 LTS VM | Mininet, Hping3, Slowloris | 192.168.56.10 | Intel core i5 Processor with 2GB RAM |

### *A. ODL Controller*

We used Linear Topology to create a network of 25 nodes using the mininet network emulator. Creating a customized topology serves no special purpose. Hence, we created a linear topology, which is easy to create and comprehend. The same results will remain valid for other types of topology. We used the hping3 tool to generate DDoS traffic [15]. This traffic is then forwarded to specific hosts of the ODL Controller. We divided our DDoS flood attack into three categories, namely low, medium and high-rate DDoS attacks.

In a low-rate DDoS attack, we only attacked node h2 using a single node h1. No significant degradation in the performance of node h1 is seen, and on average, the response time increased from around 0.5 msec to around 10 msec. Here, the Defense4All anti-DoS module thwarted the attack by redirecting the malicious traffic to other temporary nodes. When the hping3 command is stopped, the ping time of the hosts returns to its normal flow. This also shows the correct working of the Defense4All anti-DDoS module. In the medium rate DDoS attack, multiple nodes are used to attack a single node h7. Node h7 is attacked using five other hosts. The performance of node h1 degrades significantly, and ping time, on average, exceeds more than 1000 msec. When five nodes are used to attack single node h7, over 60 percent packet loss was measured from host h7.

Ten hosts were used to launch DDoS traffic in the high-rate DDoS attack. In the third phase, ten nodes (the h1 to h10) were utilized to attack node h18. Eighty-seven percent packet loss was observed, with significant deterioration in the ping time. Ping time increased to around 4000-5000 m/s. In a real-world scenario, DDoS attack traffic is generated using Botnets that contain hundreds to thousands of nodes. In that case, the ping reachability of the attacked node will stop completely, and the host will become completely dead. In all three cases, once the attack is stopped, the ODL Controller returns to its normal mode of operation. Next, we launched a Slow DDoS attack that is difficult to detect. Slow Rate DDoS attacks remain incognito for a very long time, thwarting the detection. Defense4All failed to provide any kind of protection against Slowloris, Apache killer, Slow POST DDoS, Slow Read DDoS, and low-rate DDoS attacks. Fig. 1 shows the Slowloris attack on the Simple HTTP Server created on host h1. The server was available for service only 7 percent of the time. For the remaining 93 percent of uptime, the server was filled with pending connection requests and remains unavailable for the service.

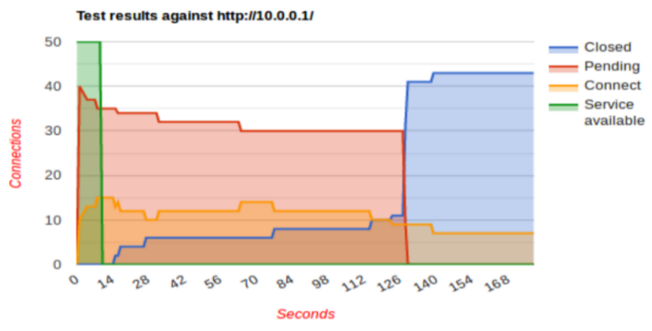


Fig. 1 The red area shows Host h1's unavailability during the Slowloris attack.

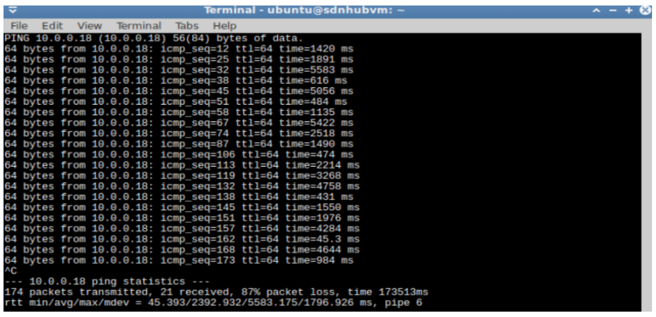
## B. *ONOS Controller*

We used two virtual machines: ONOS VM, which contains preinstalled ONOS Controller with all its necessary configurations, and the Ubuntu 16.4 VM that is used to create topology and launch an attack on ONOS Controller. Mininet network emulator is installed in Ubuntu 16.4 VM. A simple linear network topology of 10 nodes is created [16]. Topology can be seen in ONOS Tutorial VM using its GUI Application in the web browser. After that, the next step is to launch a DDoS attack on the ONOS Controller.

The Hping3 tool is used to launch the SYN Packet Flooding attack. SYN packet flood is a DoS attack that abuses the standard Three-Way Handshaking Procedure of TCP/IP Protocol. Usually, a legitimate Client sends an SYN Packet to the server to establish a connection. The server acknowledges back with SYN-ACK Packet, and finally, the Client sends the ACK packet to confirm the connection establishment. During the SYN-Flood attack, the attacker never sends the final ACK packet, so the server keeps on waiting for the Client to respond with the ACK packet. This creates many half-open TCP Connections on the server, which wastes the server limited resources and legitimate Client's need to be able to establish there a connection with the server.

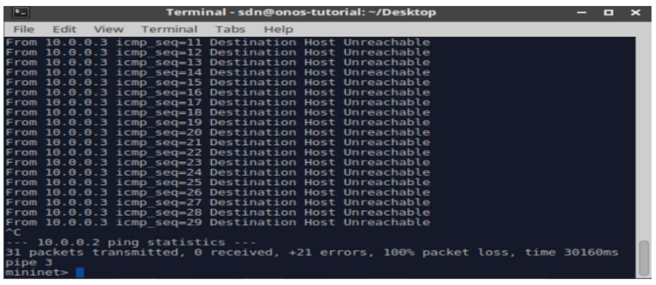
We have only performed a "low rate DDoS attack" against the ONOS Controller. Since ONOS Controller does not have any dedicated anti-DDoS mechanism, it fails to work even for a low-rate DDoS attack. To launch a DDoS attack, we used the Hping3 tool discussed above. We performed a host-to-host attack in which two hosts, h1 and h4, were used to attack node h2. Before the attack, node h3 was used to ping node h2 for network connectivity, and ping performed well during this phase. After the SYN Flood attack is launched from nodes h1 and h4, the ping from nodes h3 to h2 becomes unreachable. In simple words, node h2 stops responding to the ping command. In our experiment, we have attacked a single node, h2 using nodes h1 and h4. But, after the attack, the ping between all ten nodes becomes unreachable. Moreover, the ONOS GUI also stops responding. This shows the complete failure of the ONOS Controller against a simple SYN-Flood DDoS attack. In the real-world scenario, network bots are created to attack a server that includes thousands of hosts for attack creation. In our experiment, only two hosts were used to generate attack traffic. So, we can say ONOS Controller cannot protect against any DoS or DDoS attack in the real world.

Even when the attack is stopped, still ONOS Controller does not return to its normal flow of operation. The ping to the attacked node h2 still returns "Destination Host Unreachable". ONOS Controller is restarted manually, and then the Controller starts to work again. Unlike ODL Controller, malicious traffic redirection is not done in ONOS Controller. Even after the attack is stopped, ONOS Controller immediately does not return to its normal mode of operation. Fig. 4 shows packets captured by Wireshark Network Emulator. SYN-Flood packets are highlighted in red, and the total packets sent by hping3 in some seconds are more than 2.2 million.



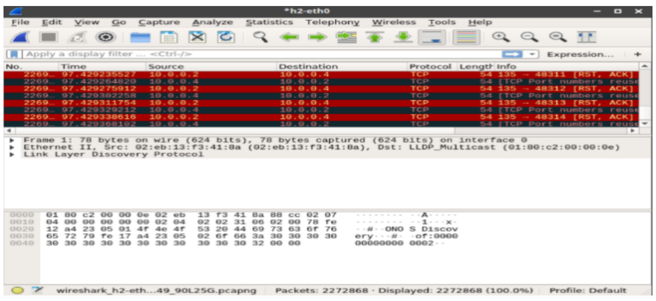
```
PING 10.0.0.18 (10.0.0.18) 56(84) bytes of data.
64 bytes from 10.0.0.18: icmp_seq=12 ttl=64 time=1420 ms
64 bytes from 10.0.0.18: icmp_seq=25 ttl=64 time=1891 ms
64 bytes from 10.0.0.18: icmp_seq=32 ttl=64 time=5583 ms
64 bytes from 10.0.0.18: icmp_seq=38 ttl=64 time=616 ms
64 bytes from 10.0.0.18: icmp_seq=45 ttl=64 time=5056 ms
64 bytes from 10.0.0.18: icmp_seq=51 ttl=64 time=484 ms
64 bytes from 10.0.0.18: icmp_seq=58 ttl=64 time=1135 ms
64 bytes from 10.0.0.18: icmp_seq=67 ttl=64 time=5422 ms
64 bytes from 10.0.0.18: icmp_seq=74 ttl=64 time=2518 ms
64 bytes from 10.0.0.18: icmp_seq=87 ttl=64 time=1490 ms
64 bytes from 10.0.0.18: icmp_seq=106 ttl=64 time=474 ms
64 bytes from 10.0.0.18: icmp_seq=113 ttl=64 time=2214 ms
64 bytes from 10.0.0.18: icmp_seq=119 ttl=64 time=3268 ms
64 bytes from 10.0.0.18: icmp_seq=132 ttl=64 time=4758 ms
64 bytes from 10.0.0.18: icmp_seq=138 ttl=64 time=431 ms
64 bytes from 10.0.0.18: icmp_seq=145 ttl=64 time=1550 ms
64 bytes from 10.0.0.18: icmp_seq=151 ttl=64 time=1976 ms
64 bytes from 10.0.0.18: icmp_seq=157 ttl=64 time=4284 ms
64 bytes from 10.0.0.18: icmp_seq=162 ttl=64 time=45.3 ms
64 bytes from 10.0.0.18: icmp_seq=168 ttl=64 time=4644 ms
64 bytes from 10.0.0.18: icmp_seq=173 ttl=64 time=984 ms
^C
--- 10.0.0.18 ping statistics ---
174 packets transmitted, 21 received, 87% packet loss, time 173513ms
rtt min/avg/max/mdev = 45.393/2392.932/5583.175/1796.926 ms, pipe 6
```

Fig. 2 87 percent packet loss for ODL Controller during TCP SYN Flood Attack.



```
From 10.0.0.3 icmp_seq=11 Destination Host Unreachable
From 10.0.0.3 icmp_seq=12 Destination Host Unreachable
From 10.0.0.3 icmp_seq=13 Destination Host Unreachable
From 10.0.0.3 icmp_seq=14 Destination Host Unreachable
From 10.0.0.3 icmp_seq=15 Destination Host Unreachable
From 10.0.0.3 icmp_seq=16 Destination Host Unreachable
From 10.0.0.3 icmp_seq=17 Destination Host Unreachable
From 10.0.0.3 icmp_seq=18 Destination Host Unreachable
From 10.0.0.3 icmp_seq=19 Destination Host Unreachable
From 10.0.0.3 icmp_seq=20 Destination Host Unreachable
From 10.0.0.3 icmp_seq=21 Destination Host Unreachable
From 10.0.0.3 icmp_seq=22 Destination Host Unreachable
From 10.0.0.3 icmp_seq=23 Destination Host Unreachable
From 10.0.0.3 icmp_seq=24 Destination Host Unreachable
From 10.0.0.3 icmp_seq=25 Destination Host Unreachable
From 10.0.0.3 icmp_seq=26 Destination Host Unreachable
From 10.0.0.3 icmp_seq=27 Destination Host Unreachable
From 10.0.0.3 icmp_seq=28 Destination Host Unreachable
From 10.0.0.3 icmp_seq=29 Destination Host Unreachable
^C
--- 10.0.0.2 ping statistics ---
31 packets transmitted, 0 received, +21 errors, 100% packet loss, time 30160ms
pipe 3
mininet>
```

Fig. 3 100 percent packet loss for ONOS Controller during TCP SYN Flood



```
No.       Time          Source     Destination  Protocol Length Info
2269...   97.429235527  10.0.0.2   10.0.0.4     TCP      54 135 → 48311 [RST, ACK]
2269...   97.429264820  10.0.0.4   10.0.0.2     TCP      54 [TCP Port numbers reuse
2269...   97.429275912  10.0.0.2   10.0.0.4     TCP      54 135 → 48312 [RST, ACK]
2269...   97.429302258  10.0.0.4   10.0.0.2     TCP      54 [TCP Port numbers reuse
2269...   97.429311754  10.0.0.2   10.0.0.4     TCP      54 135 → 48313 [RST, ACK]
2269...   97.429329212  10.0.0.4   10.0.0.2     TCP      54 [TCP Port numbers reuse
2269...   97.429338616  10.0.0.2   10.0.0.4     TCP      54 135 → 48314 [RST, ACK]
2269...   97.429368102  10.0.0.4   10.0.0.2     TCP      54 [TCP Port numbers reuse

▸ Frame 1: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
▸ Ethernet II, Src: 02:eb:13:f3:41:8a (02:eb:13:f3:41:8a), Dst: LLDP_Multicast (01:80:c2:00:00:0e)
▸ Link Layer Discovery Protocol

0000  01 80 c2 00 00 0e 02 eb  13 f3 41 8a 88 cc 02 07   ········ ··A·····
0010  04 00 00 00 00 00 02 04  02 02 31 06 02 00 78 fe   ········ ··1···x·
0020  12 a4 23 05 01 4f 4e 4f  53 20 44 69 73 63 6f 76   ··#··ONO S Discov
0030  65 72 79 fe 17 a4 23 05  02 6f 66 3a 30 30 30 30   ery···#· ·of:0000
0040  30 30 30 30 30 30 30 30  30 30 30 32 00 00         00000000 0002··

wireshark_h2-eth...49_90L25G.pcapng   Packets: 2272868 · Displayed: 2272868 (100.0%)   Profile: Default
```

Fig. 4 TCP SYN Flood packets are highlighted in red by Wireshark.

## VI. Conclusion

Single Point of failure of the Control layer in SDN Controller is a well-known issue. Still, ODL and ONOS Controller suffer from the same. ODL Controller has tried to mitigate the DDOS attack by deploying a custom anti-DDoS mechanism known as “Defense4All’. But unfortunately, Defense4All can mitigate only low-rate DDoS attacks and fails to protect the ODL Controller against medium and high-rate attacks. ONOS Controller has no anti-DDoS mechanism, and the ONOS Controller cannot withstand even a low-rate DDoS attack. This paper discussed the security issue that needs to be addressed by the SDN developers. We found out that ODL Controller is more immune than the ONOS Controller. For future research, we are working on an anti-DDoS mechanism that can be deployed for any SDN Controller.

## REFERENCES

[1] How many iot devices are there in 2022?, https://techjury.net/blog/how-manyiot-devices-are-there/#gref Accessed on: 12-12-2022.

[2] Difference between software defined network and traditional network, https://www.geeksforgeeks.org/difference-between-software -defined-networkand-traditional-network/. Accessed on: 12-12-2022.

[3] N. Feamster, J. Rexford, and E. Zegura, “The road to SDN,” ACM SIGCOMM Computer Communication Review, vol. 44, no. 2. Association for Computing Machinery (ACM), pp. 87–98, Apr. 08, 2014. doi: 10.1145/2602204.2602219.

[4] E. Zeydan and Y. Turk, “Recent Advances in Intent-Based Networking: A Survey,” 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring). IEEE, May 2020. doi: 10.1109/vtc2020-spring48590.2020.9128422.

[5] Why is intent-based networking good news for software-defined networking? URL https: //blogs.cisco.com/analytics-automation/w hy-is-intent-based-networking-good-newsfor-software-defined-networking. Accessed on: 12-12-2022.

[6] Y. Han, J. Li, D. Hoang, J.-H. Yoo, and J. W.-K. Hong, “An intent-based network virtualization platform for SDN,” 2016 12th International Conference on Network and Service Management (CNSM). IEEE, Oct. 2016. doi: 10.1109/cnsm.2016.7818446.

[7] F. Neto, A. Santos, C. Jacob Miguel, and P. Sampaio, “SDN Controllers -A Comparative approach to Market Trends”, In 9th International Workshop on ADVANCEs in ICT Infrastructures and Services (ADVANCE 2021), 02 2021.

[8] J. Singh and S. Behal, “Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions,” Computer Science Review, vol. 37. Elsevier BV, p. 100279, Aug. 2020. doi: 10.1016/j.cosrev.2020.100279.

[9] Radware releases defense4all, industry-first open sdn security application for opendaylight project. URL https://www.radware.co m/newsevents/pressreleases/radware-relea ses-defense4all-industry-first-open-sdn-secu rity-applicationfor-opendaylight-project/. Accessed on: 12-12-2022.

[10] 104238 - opendaylight controller ’sdnidatabase.java’ sql injection vulnerability(2018-05-19). URL https: //www.cvedetails.com/bugtraq-bid/104238/OpenDaylight-Controller-SdniDataBa se.java-SQL-Injection-Vu.html. Accessed on: 12-12-2022.

[11] Security-mode onos. URL http://events17.l inuxfoundation.org/sites/events/files/slides/ smonos ons2016.pdf. Accessed on: 12-12-2022.

[12] Secci, S., Attou, K., Phung, C. D., Scott-Hayward, S., Vemuri, S., & Wang, Y. (2017). ONOS Security & Performance Analysis (Report No. 1) (Doctoral dissertation, ONOS).

[13] R. L. S. de Oliveira, C. M. Schweitzer, A. A. Shinoda, and Ligia Rodrigues Prete, “Using Mininet for emulation and prototyping Software-Defined Networks,” 2014 IEEE Colombian Conference on Communications and Computing (COLCOM). IEEE, Jun. 2014. doi: 10.1109/colcomcon.2014.6860404.

[14] Dash, P. (2013). Getting started with oracle vm virtualbox. Packt Publishing.

[15] L. Liang, K. Zheng, Q. Sheng and X. Huang, "A Denial of Service Attack Method for an IoT System," 2016 8th International Conference on Information Technology in Medicine and Education (ITME), 2016, pp. 360-364, doi: 10.1109/ITME.2016.0087.

[16] R. L. S. de Oliveira, C. M. Schweitzer, A. A. Shinoda and Ligia Rodrigues Prete, "Using Mininet for emulation and prototyping Software-Defined Networks," 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), 2014, pp. 1-6, doi: 10.1109/ColComCon.2014.6860404.